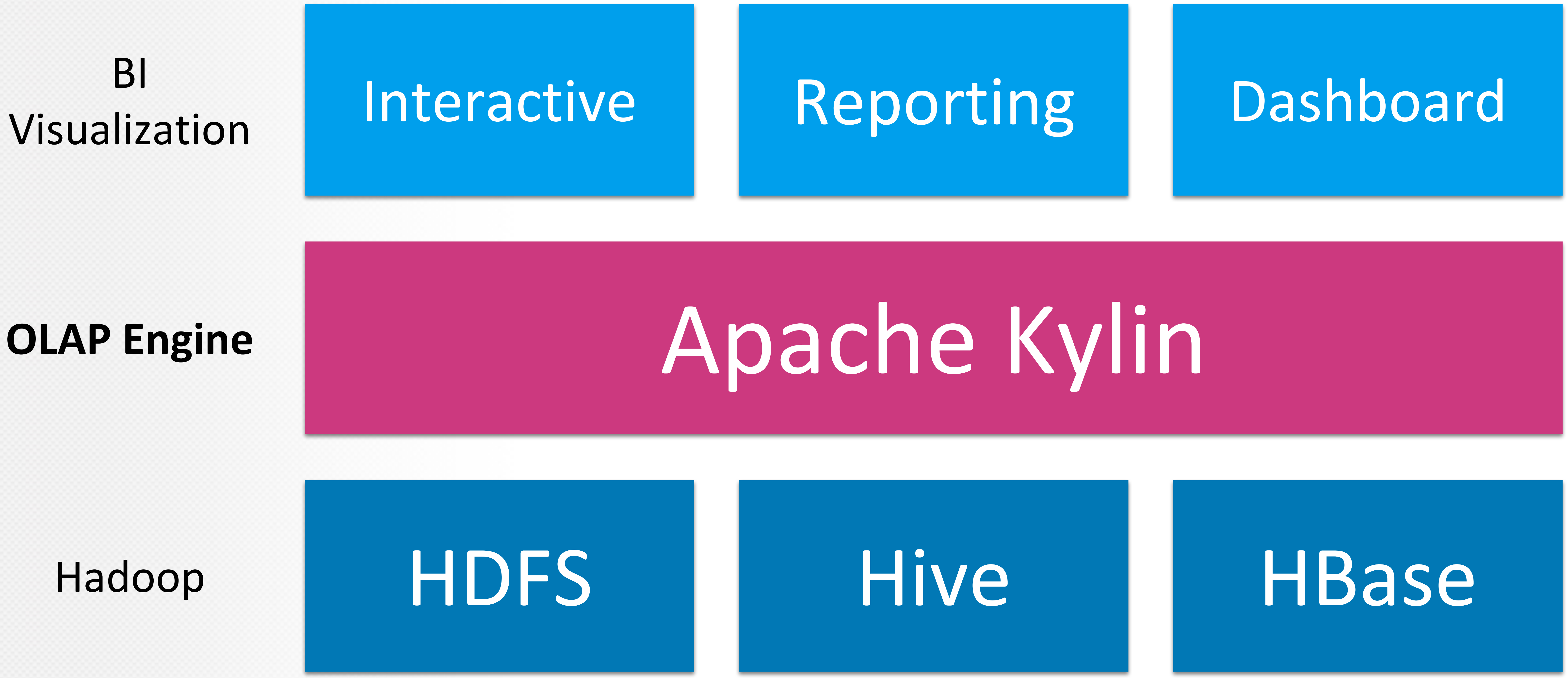
Apache Kylin v2.0 最新功能和深度技术解读

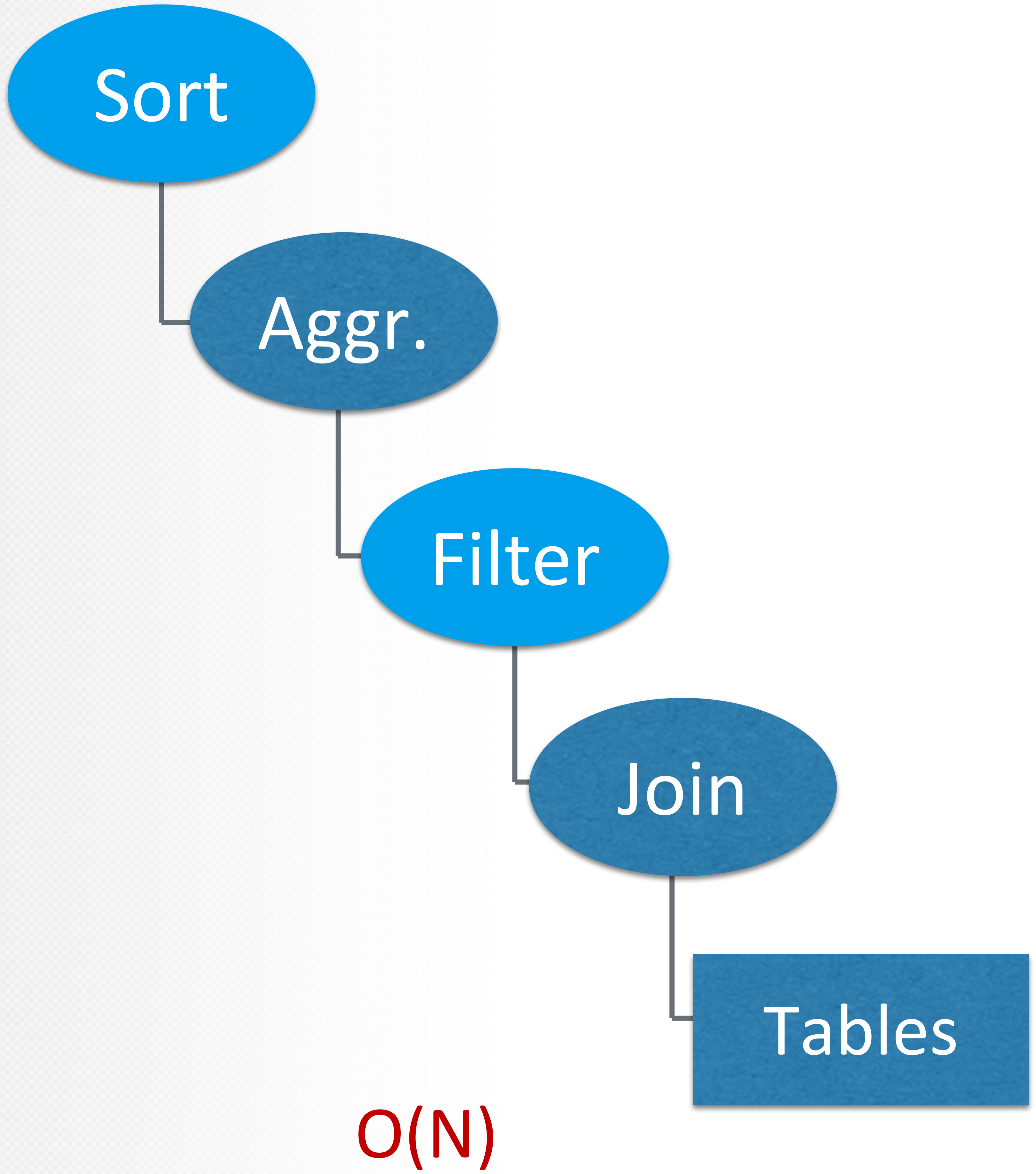
李扬 | Li, Yang
Co-founder & CTO

Apache Kylin是什么



- 3万亿行, 查询时间 < 1 sec
  @toutiao.com, top 1 news feed app in China
- 60+ 维度 cube
  @CPIC, top 3 insurance group in China
- JDBC / ODBC / RestAPI
- BI 无缝集成

Kylin 为什么快

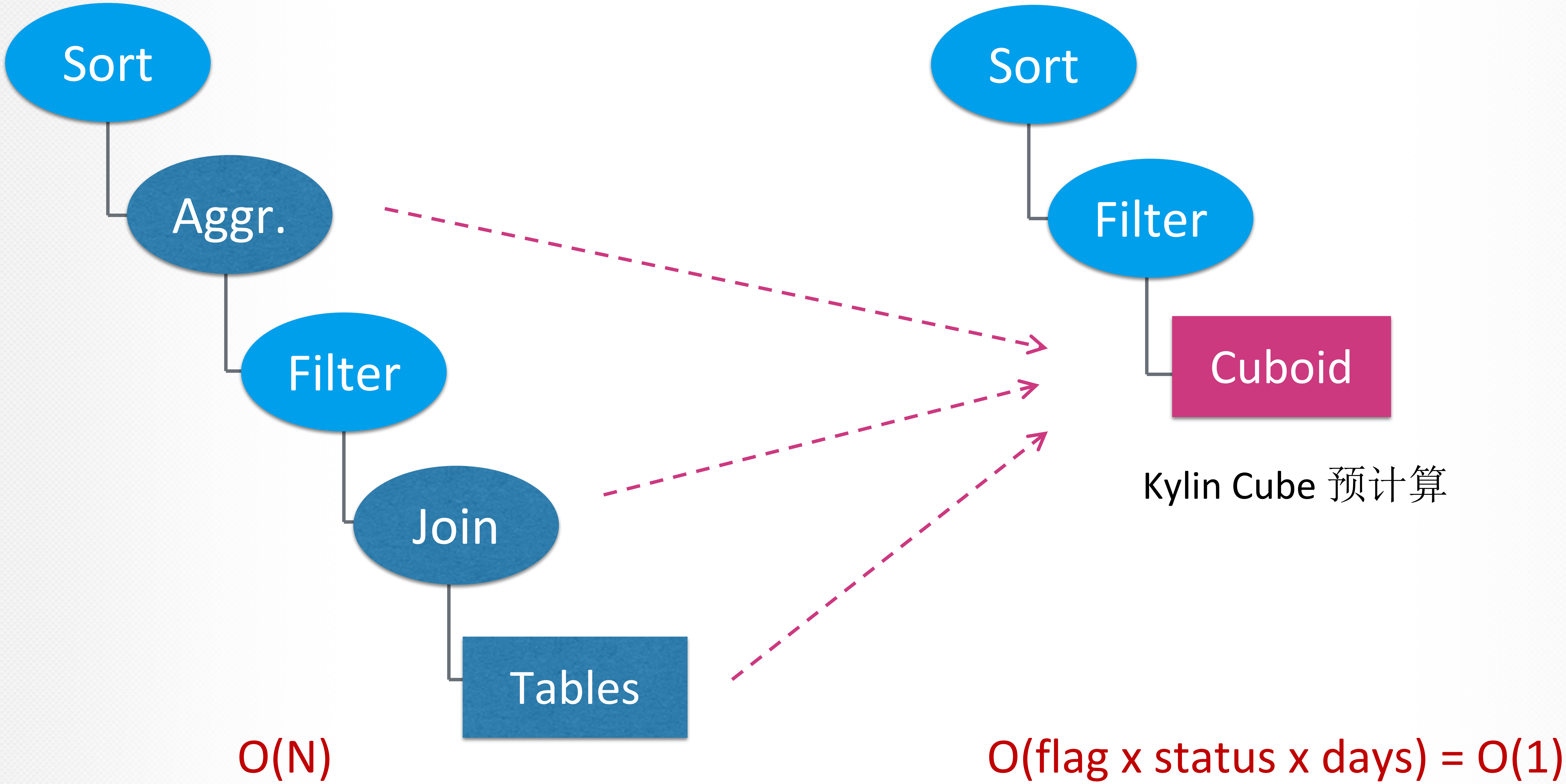


一个例子: 给定时间范围，按 “returnflag” 和 “orderstatus” 报告营收。

```
select
   l_returnflag,
   o_orderstatus,
   sum(l_quantity) as sum_qty,
   sum(l_extendedprice) as sum_base_price
   …
from
   v_lineitem
   inner join v_orders on l_orderkey = o_orderkey
where
   l_shipdate <= '1998-09-16'
group by
   l_returnflag,
   o_orderstatus
order by
   l_returnflag,
   o_orderstatus;
```

Kyligence

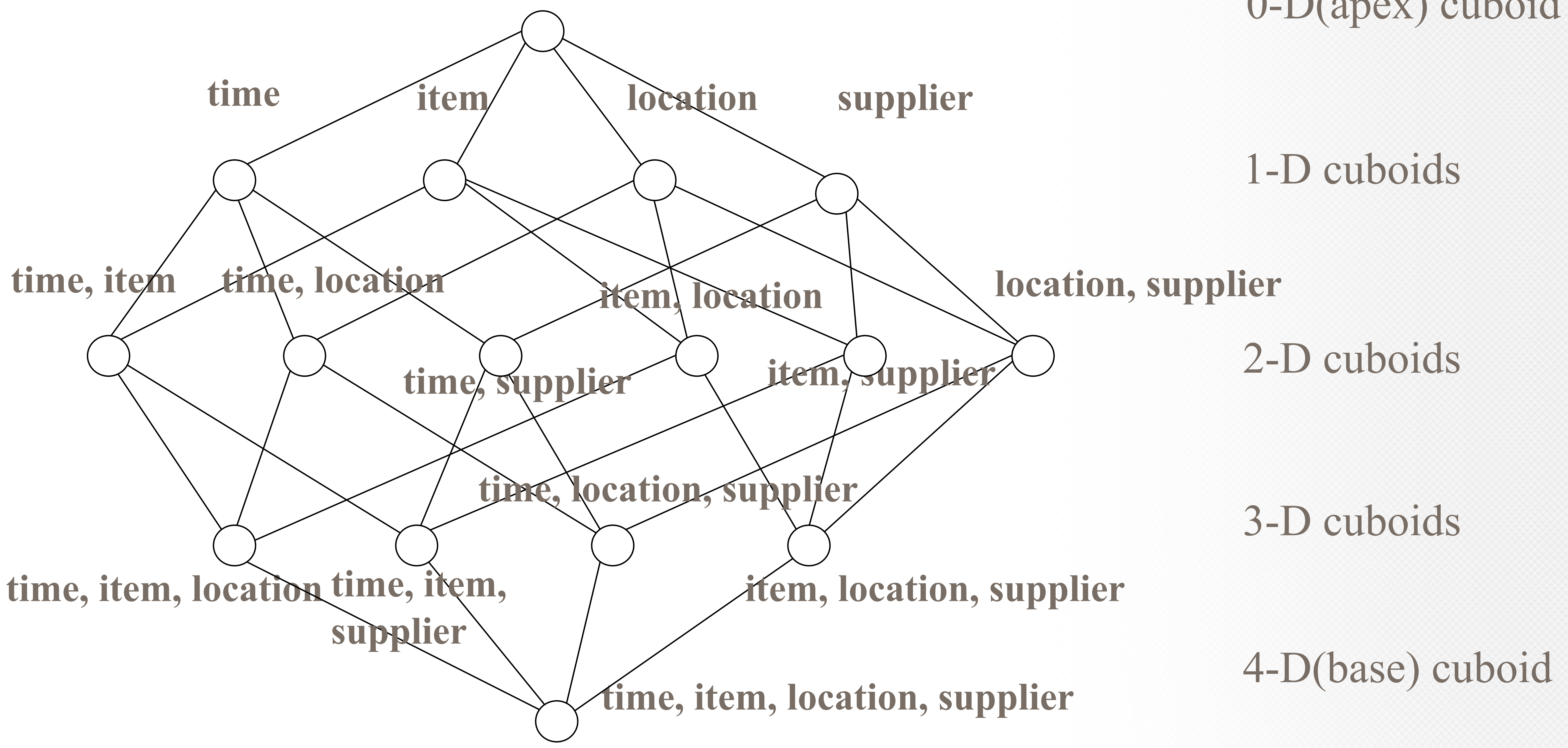
Kylin 为什么快



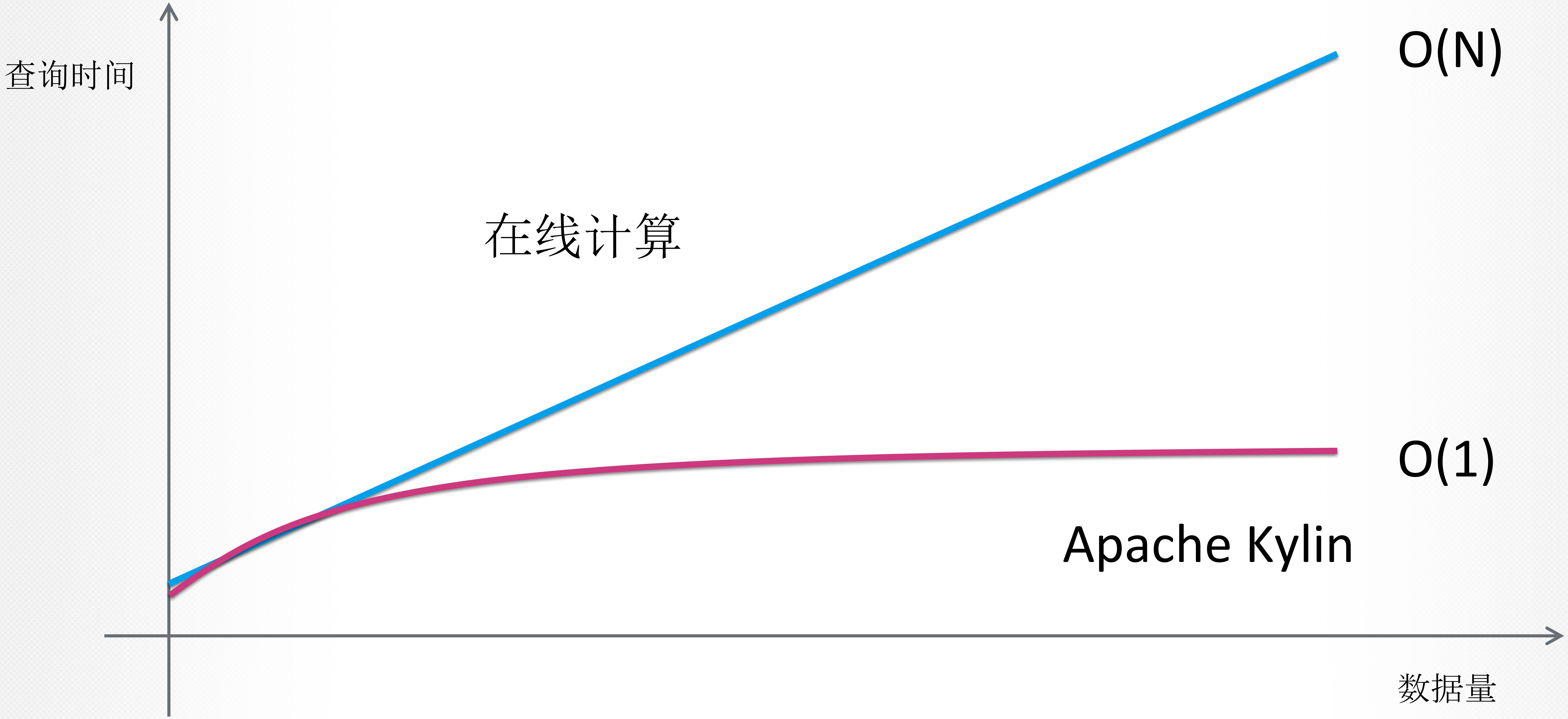
Sort
Aggr.
Filter
Join
Tables
Sort
Filter
Cuboid
Kylin Cube 预计算
O(N)
O(flag x status x days) = O(1)

Kylin 就是预计算

- 基于 Cube 理论
- Model 和 Cube 定义了预计算空间
- Build Engine 执行预计算
- Query Engine 在预计算结果上执行 SQL



O(1) 无论数据多大

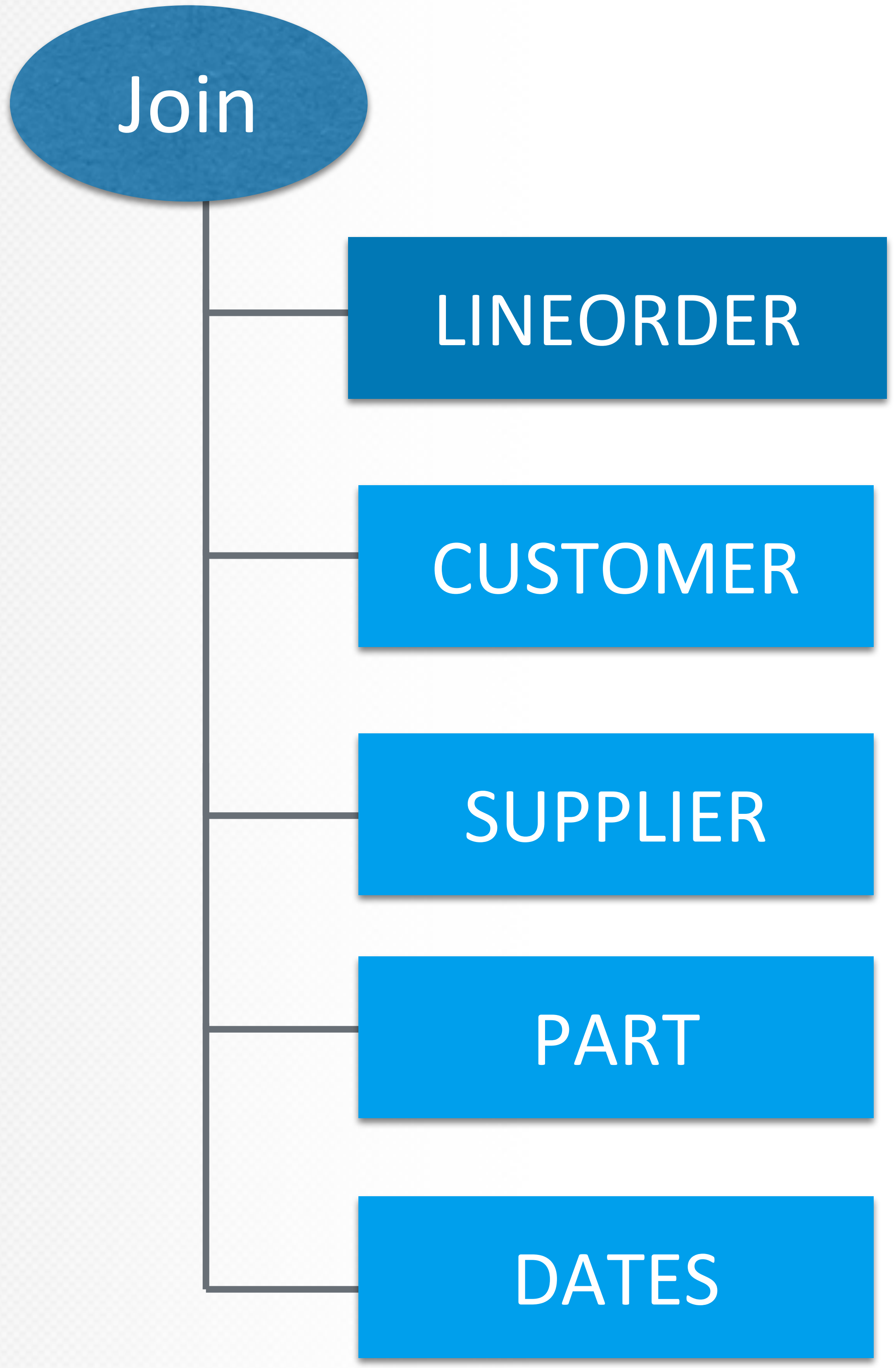


Kyligence

支持雪花模型

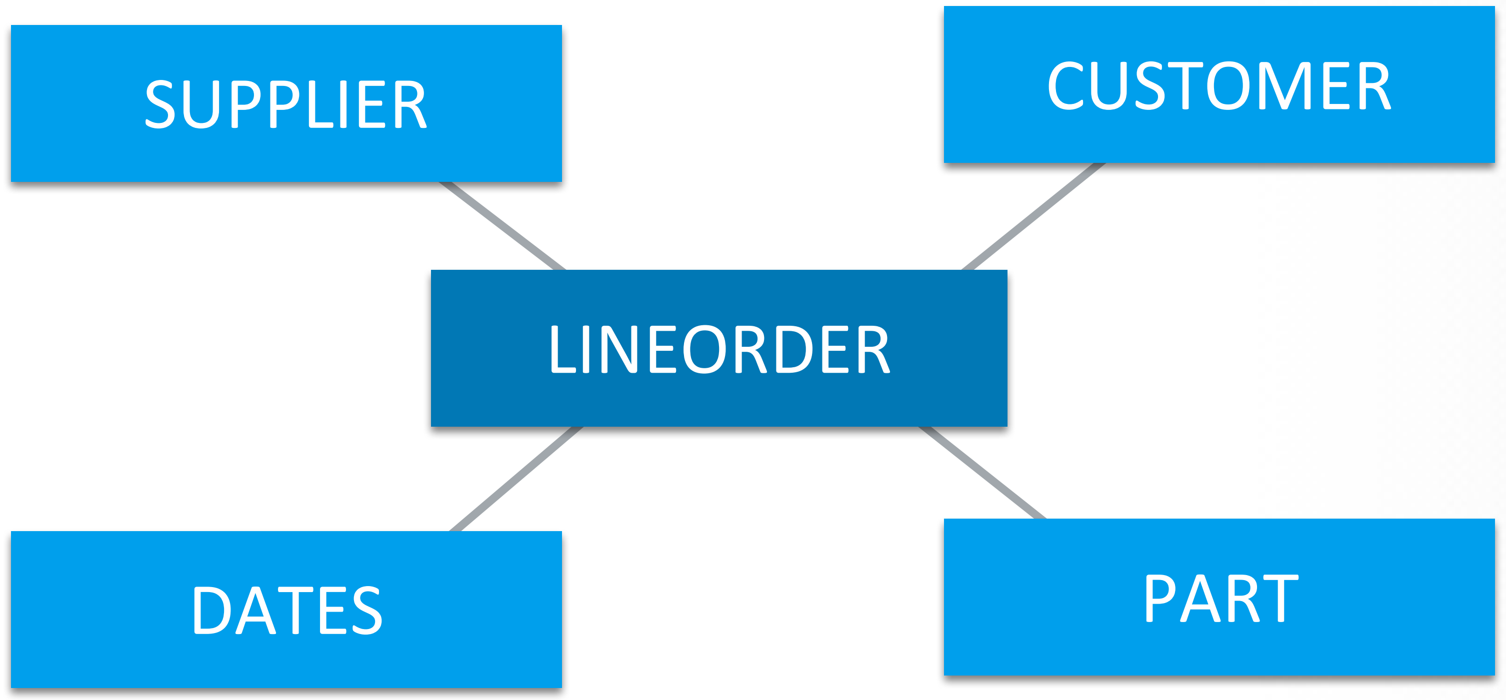
运行 TPC-H 基准测试

Kylin 1.0 受限于星形模型



只支持一层维表

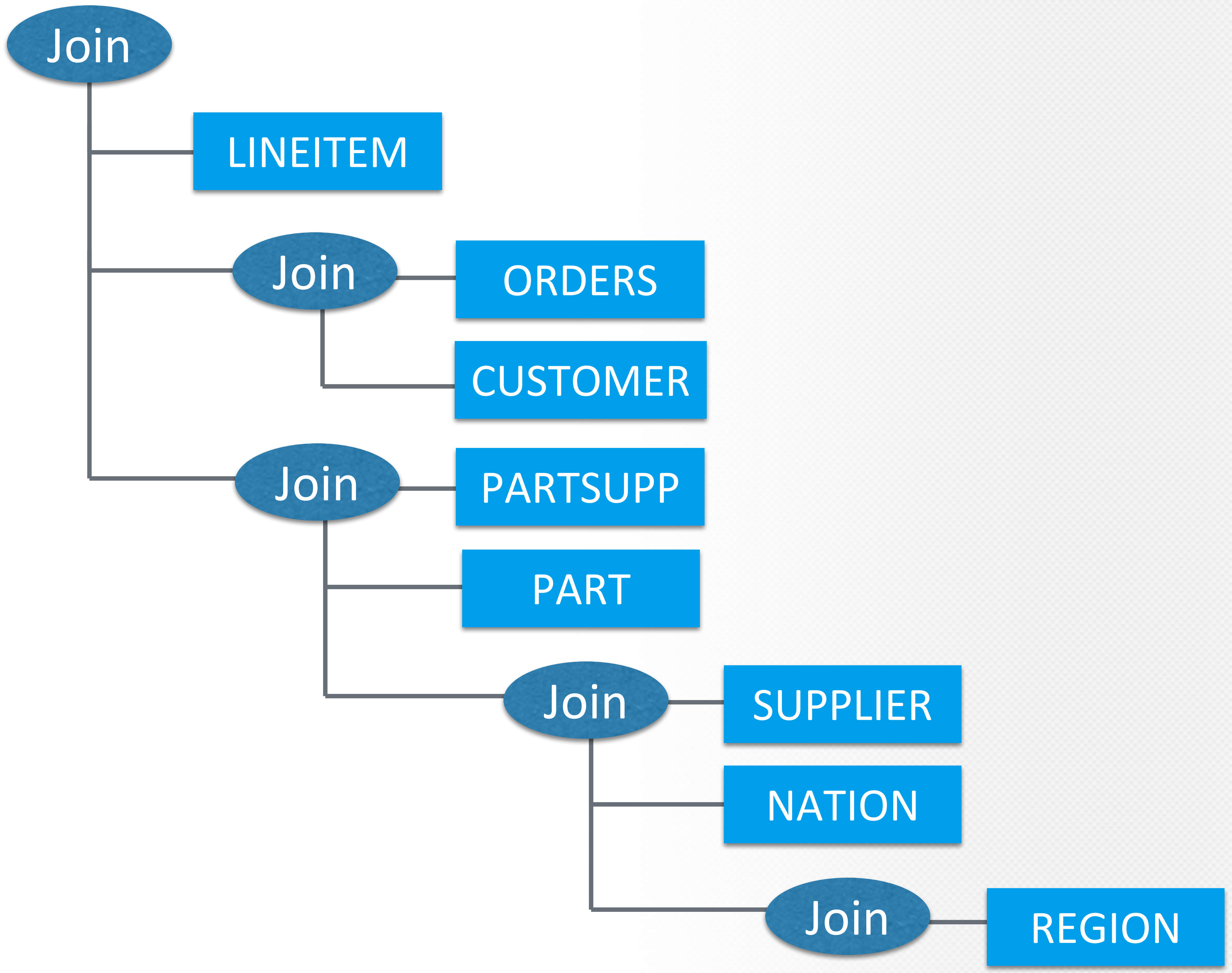
- 只支持星形模型
- 不允许同名列即使在不同的表上
- 不允许一张表关联自身
- 无法直接支持真实的业务场景



Kylin 2.0 雪花模型

支持多层维表

- 支持雪花模型 (KYLIN-1875)
- 支持同一张表被关联多次
- Model 的元数据有较大变化
- 修复了很多关于子查询和多表关联的缺陷
- 支持任意复杂的模型，任意灵活的查询



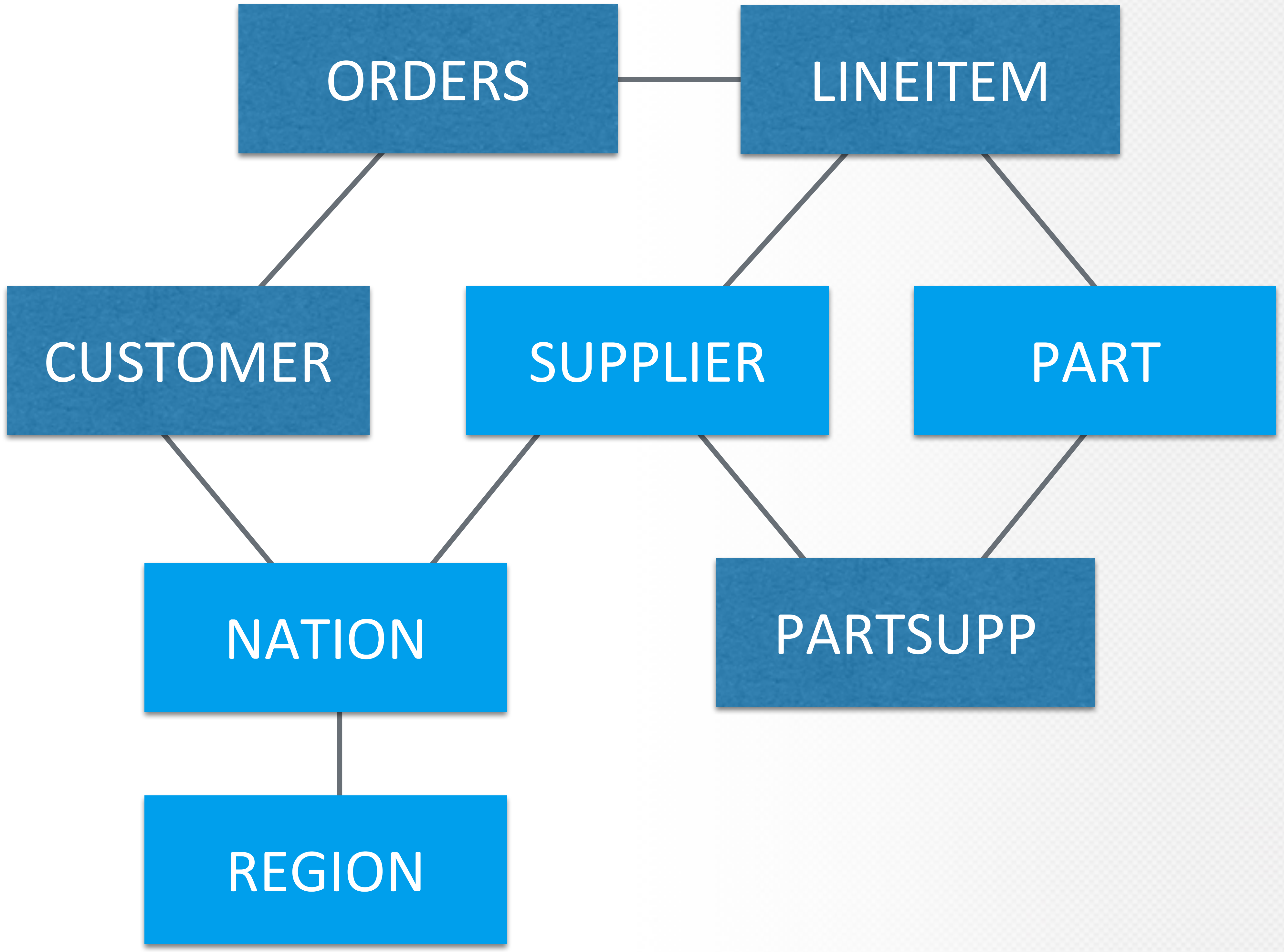
Kylin 2.0 上的 TPC-H 测试

TPC-H 是一个决策支持系统的基准测试

- 流行于商业 RDBMS 和 Data Warehouse 系统
- 查询和数据模型有广泛的行业普适性
- 分析大量数据
- 执行高复杂度的查询
- 回答关键的业务问题

Kylin 2.0 可以执行所有 22 条 TPC-H 查询 (KYLIN-2467)

- 通过预计算可以执行非常灵活和复杂的查询
- SQL 兼容性是现阶段的目标，而不是性能
- 具体步骤: https://github.com/Kyligence/kylin-tpch

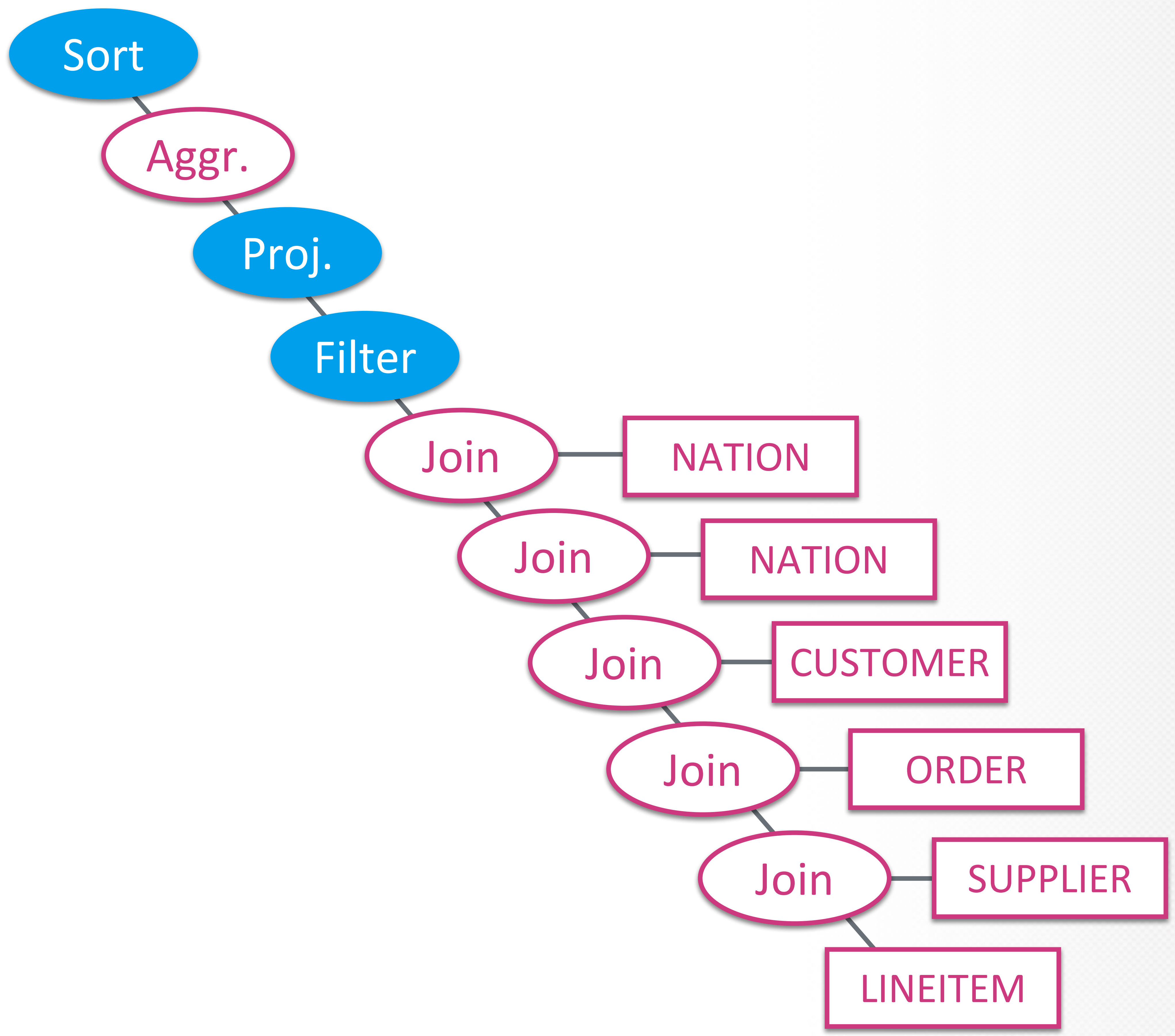


复杂查询 1

TPC-H query 07

- 0.17 sec (Hive+Tez 35.23 sec)
- 2 sub-queries

```
select
  supp_nation,
  cust_nation,
  l_year,
  sum(volume) as revenue
from
  (
    select
      n1.n_name as supp_nation,
      n2.n_name as cust_nation,
      l_shipyear as l_year,
      l_saleprice as volume
    from
      v_lineitem
      inner join supplier on s_suppkey = l_suppkey
      inner join v_orders on l_orderkey = o_orderkey
      inner join customer on o_custkey = c_custkey
      inner join nation n1 on s_nationkey = n1.n_nationkey
      inner join nation n2 on c_nationkey = n2.n_nationkey
    where
      (
        (n1.n_name = 'KENYA' and n2.n_name = 'PERU')
        or (n1.n_name = 'PERU' and n2.n_name = 'KENYA')
      )
      and l_shipdate between '1995-01-01' and '1996-12-31'
  ) as shipping
group by
  supp_nation,
  cust_nation,
  l_year
order by
  supp_nation,
  cust_nation,
  l_year
```



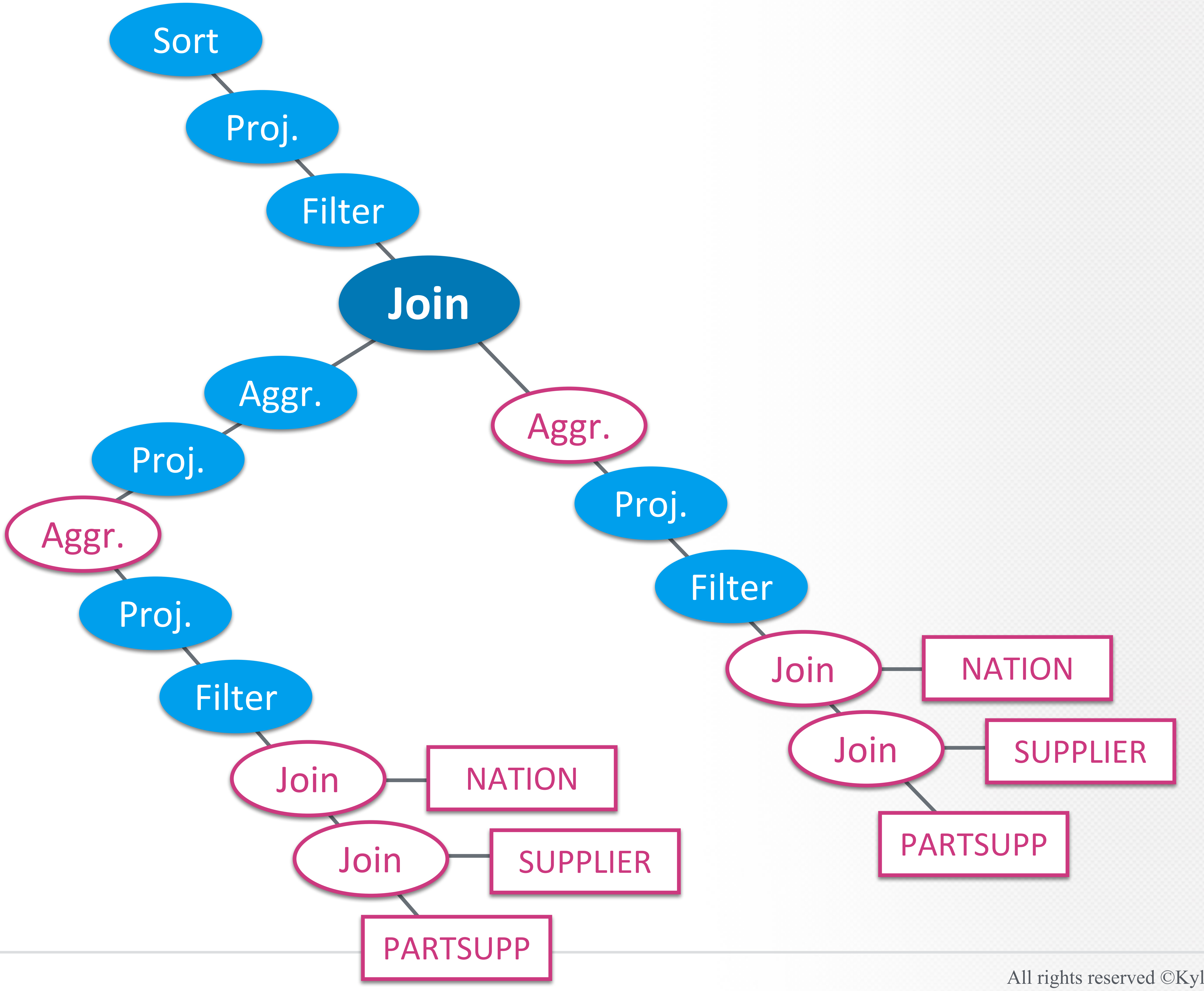
复杂查询 2

TPC-H query 11

- 3.42 sec (Hive+Tez 15.87 sec)
- 4 sub-queries, 1 online join

```
with q11_part_tmp_cached as (
  select
    ps_partkey,
    sum(ps_partvalue) as part_value
  from
    v_partsupp
    inner join supplier on ps_suppkey = s_suppkey
    inner join nation on s_nationkey = n_nationkey
  where
    n_name = 'GERMANY'
  group by ps_partkey
),
q11_sum_tmp_cached as (
  select
    sum(part_value) as total_value
  from
    q11_part_tmp_cached
)

select
  ps_partkey,
  part_value
from (
  select
    ps_partkey,
    part_value,
    total_value
  from
    q11_part_tmp_cached, q11_sum_tmp_cached
) a
where
  part_value > total_value * 0.0001
order by
  part_value desc;
```

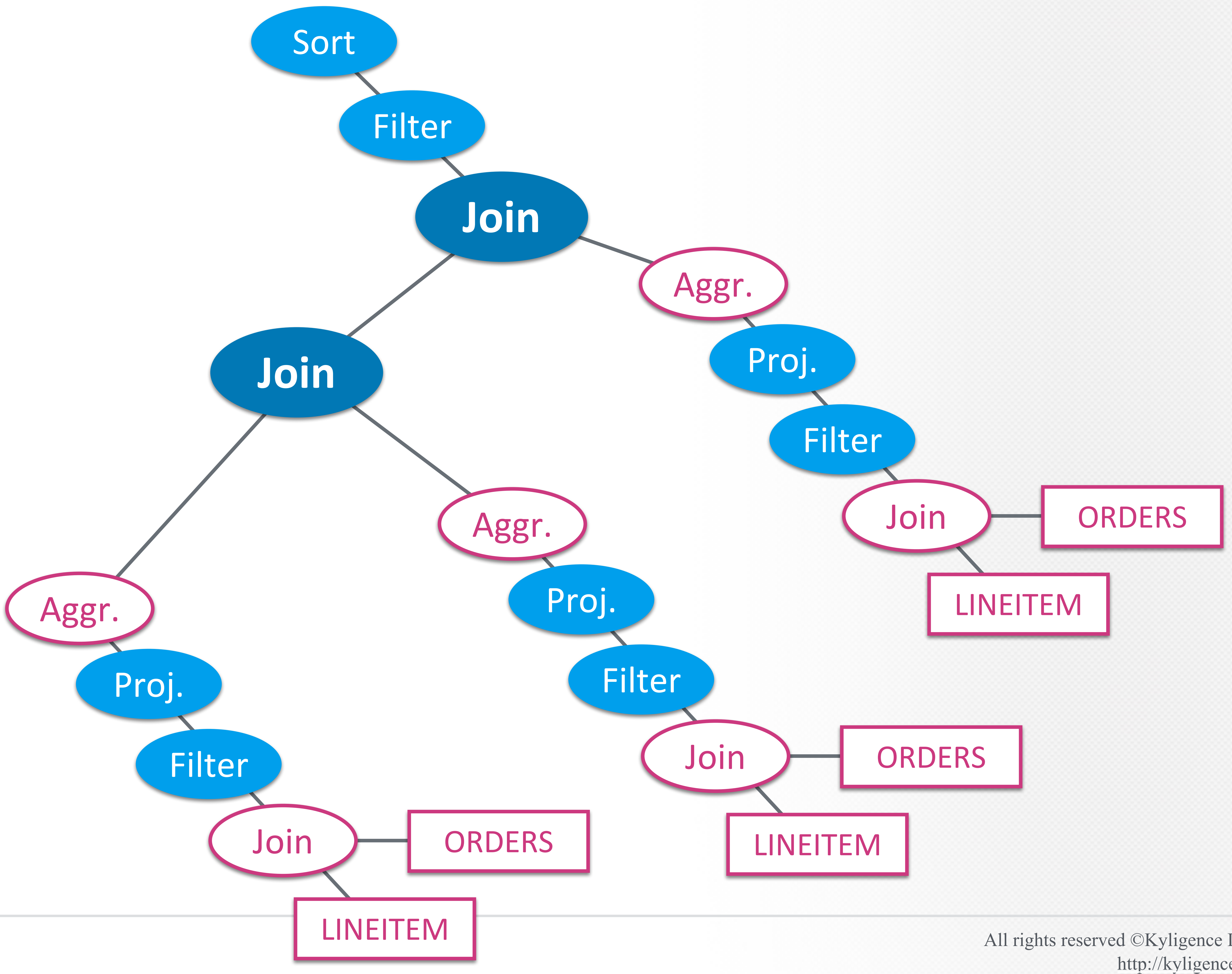


复杂查询 3

TPC-H query 12

- 7.66 sec (Hive+Tez 12.64 sec)
- 5 sub-queries, 2 online joins

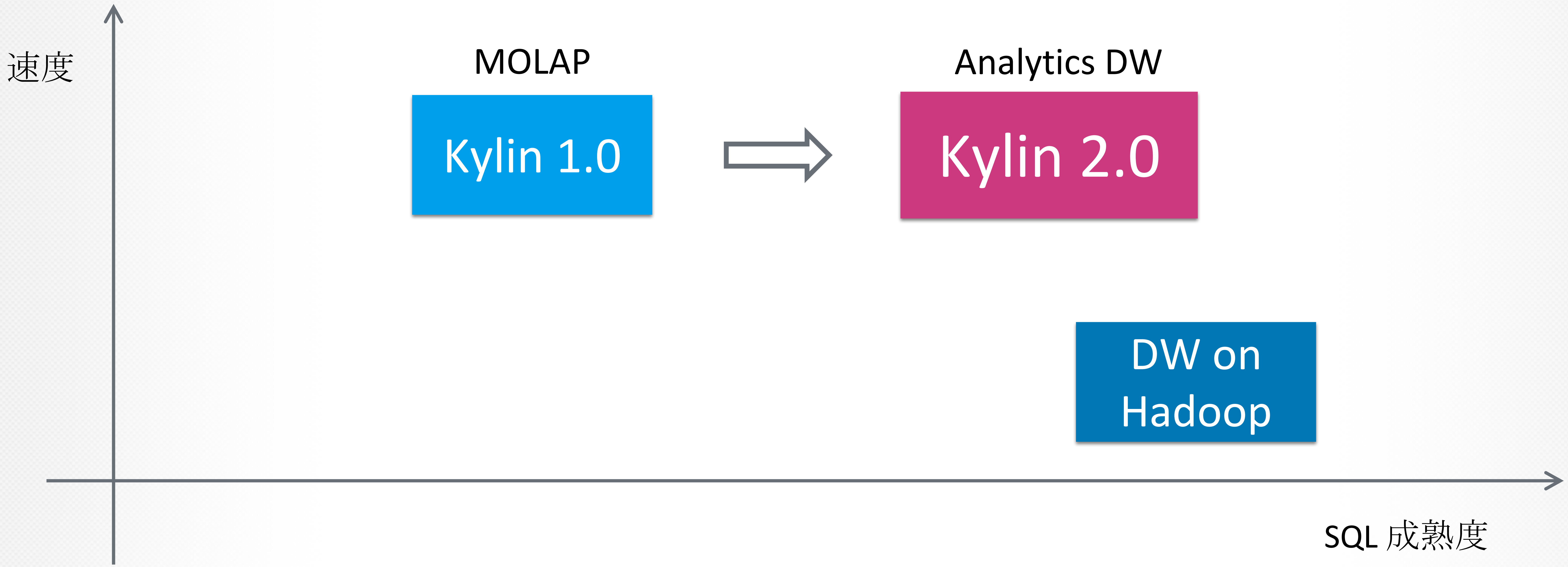
```
with in_scope_data as(
  select
    l_shipmode,
    o_orderpriority
  from
    v_lineitem inner join v_orders on l_orderkey = o_orderkey
  where
    l_shipmode in ('REG AIR', 'MAIL')
    and l_receiptdelayed = 1
    and l_shipdelayed = 0
    and l_receiptdate >= '1995-01-01'
    and l_receiptdate < '1996-01-01'
), all_l_shipmode as(
  select
    distinct l_shipmode
  from
    in_scope_data
), high_line as(
  select
    l_shipmode,
    count(*) as high_line_count
  from
    in_scope_data
  where
    o_orderpriority = '1-URGENT' or o_orderpriority = '2-HIGH'
  group by l_shipmode
), low_line as(
  select
    l_shipmode,
    count(*) as low_line_count
  from
    in_scope_data
  where
    o_orderpriority <> '1-URGENT' and o_orderpriority <> '2-HIGH'
  group by l_shipmode
)
select
  al.l_shipmode, hl.high_line_count, ll.low_line_count
from
  all_l_shipmode al
  left join high_line hl on al.l_shipmode = hl.l_shipmode
  left join low_line ll on al.l_shipmode = ll.l_shipmode
order by
  al.l_shipmode
```



Kyligence

超越 MOLAP

- 支持复杂模型和子查询; TPC-H 兼容
- Percentile / Window / Time 函数



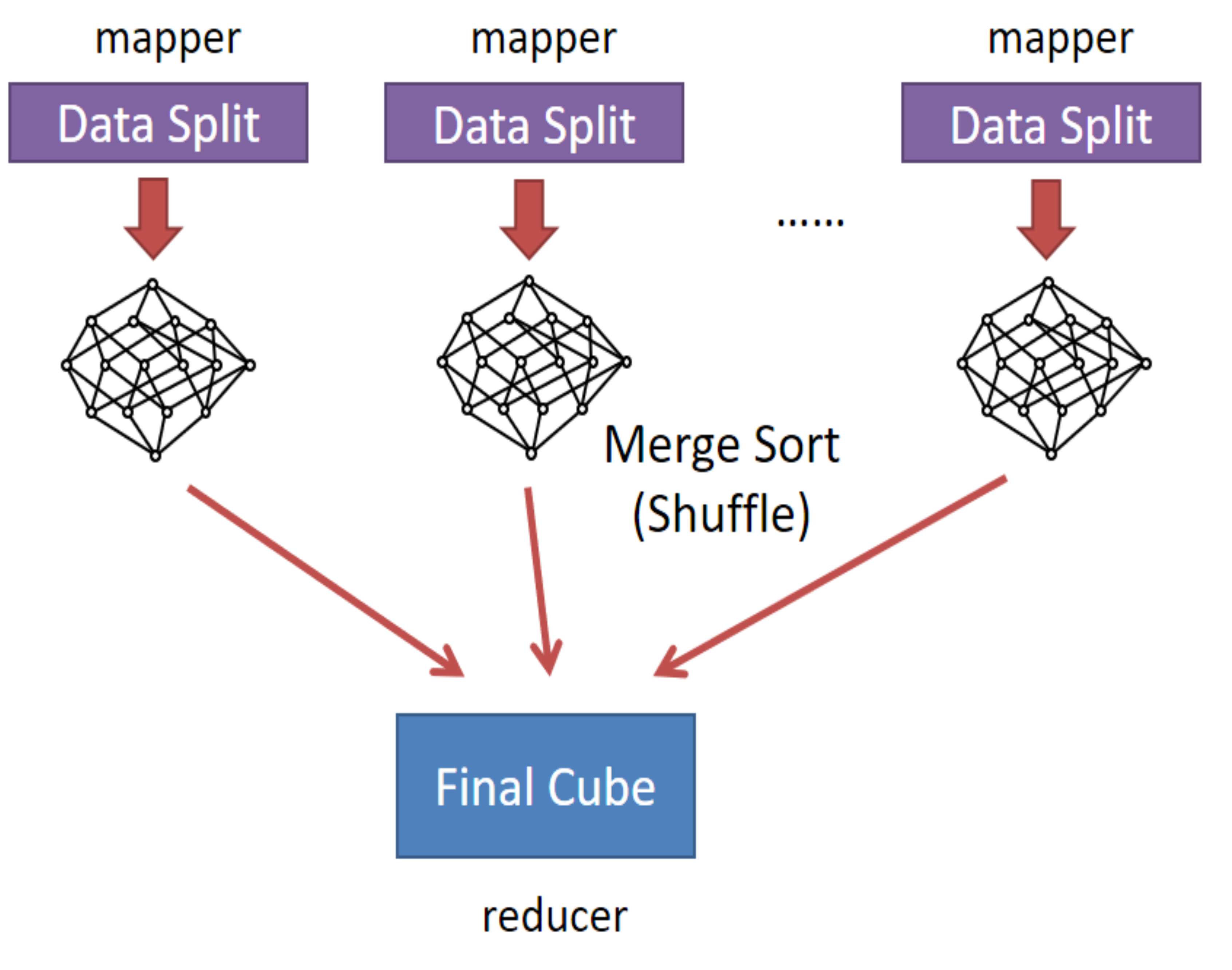
Spark Cubing

构建时间减半

一点历史

Kylin 1.5 就尝试过 Spark 构建，但这个功能从未发布

- 移植了 MR in-mem cubing 构建算法
- 占用内存，用一轮 MR 完成全部 Cube 构建
- 没有明显的性能提升
- 因为这里 Spark 与 MR 并没有什么不同

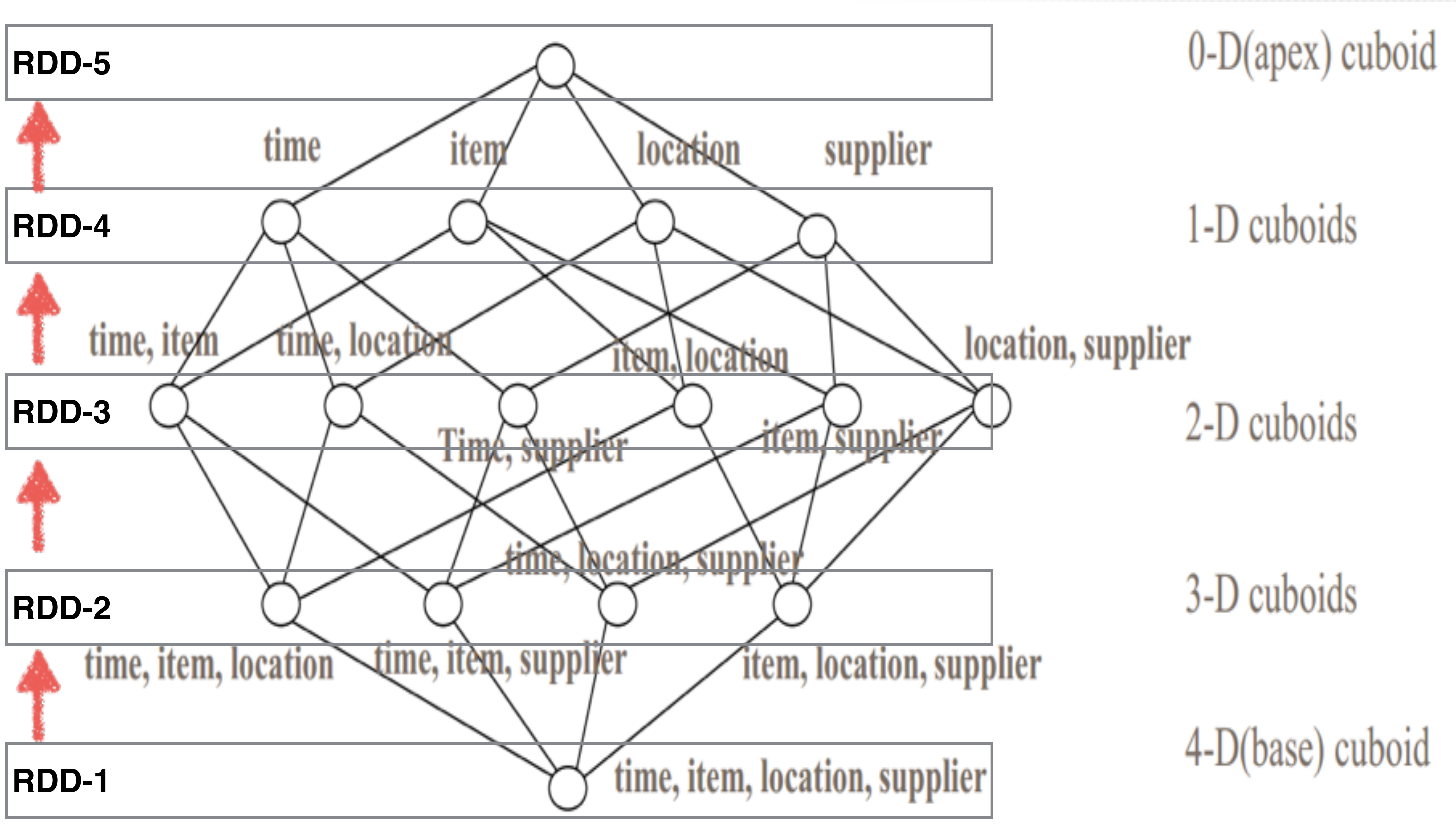


Kyligence

2.0 中的 Spark Cubing

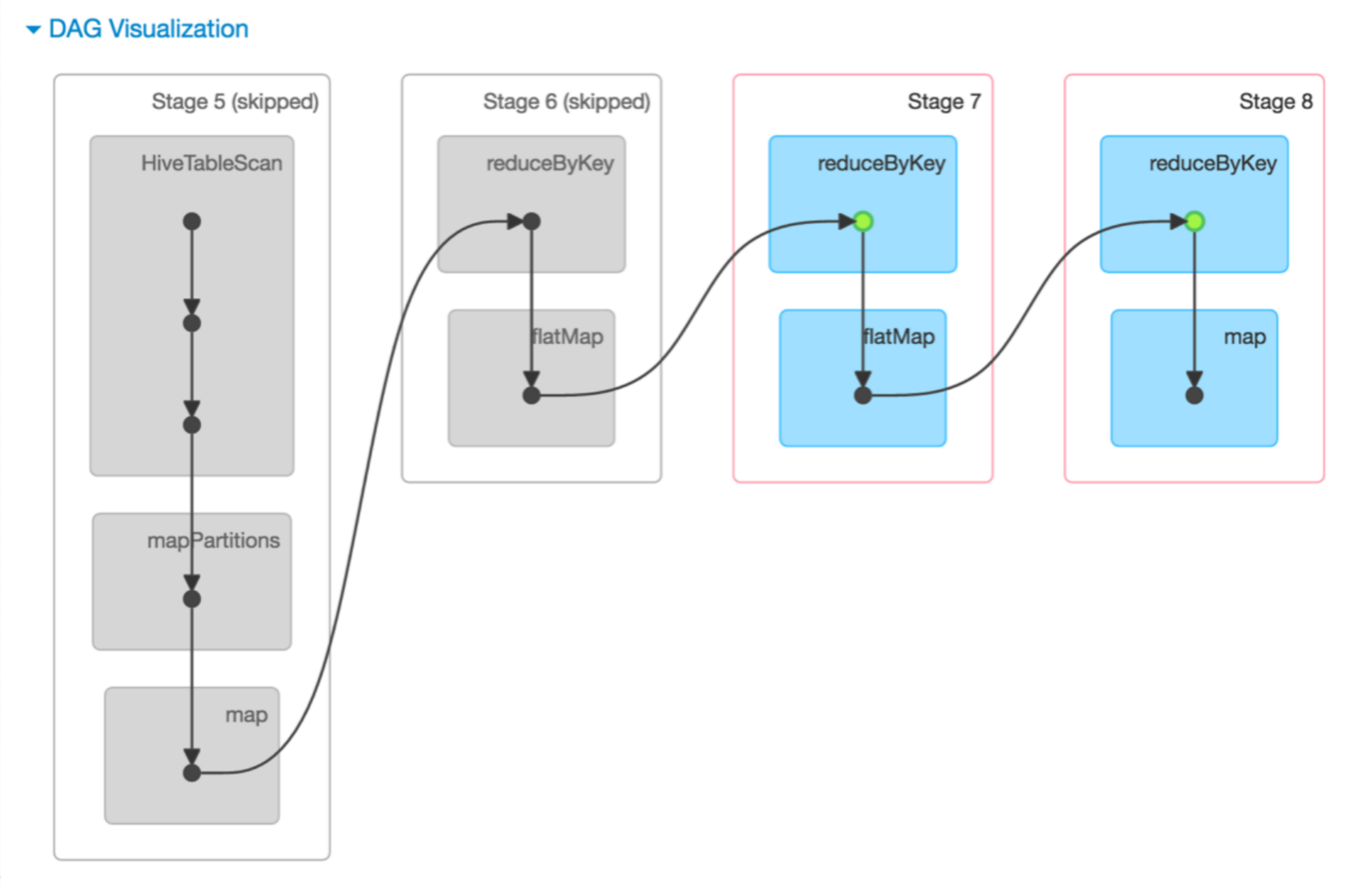
Kylin 2.0 基于 Layered Cubing 构建算法重新实现了 Spark 构建引擎。

- 每一层 cuboids 是一个 RDD
- 前继 RDD 被缓存来加速下一轮
- RDD 最终被保存为 sequence file, 格式与 MR 兼容
- 重用代码， “map” to “flatMap”; “reduce” to “reduceByKey”



Kyligence

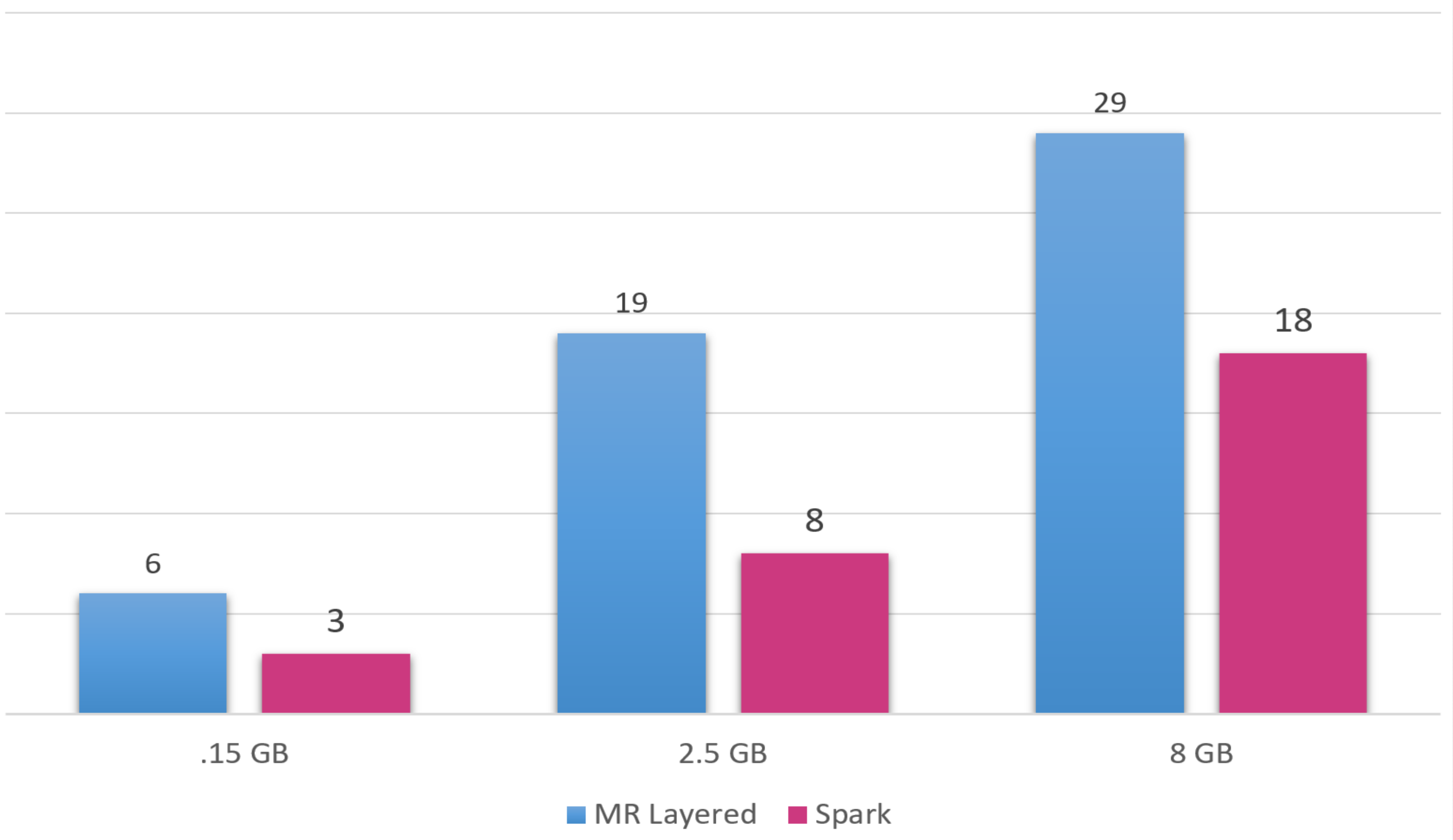
例子：计算第3层的 DAG



Spark Cubing vs. MR Layered Cubing

构建时间减半。优势随着数据量的增加而缩小。

- 4-node cluster
- Spark 1.6.3 on YARN
- 24 vcores, 30 GB memory
- 测试 3 个不断增大的数据集
  .15 GB / 2.5 GB / 8 GB

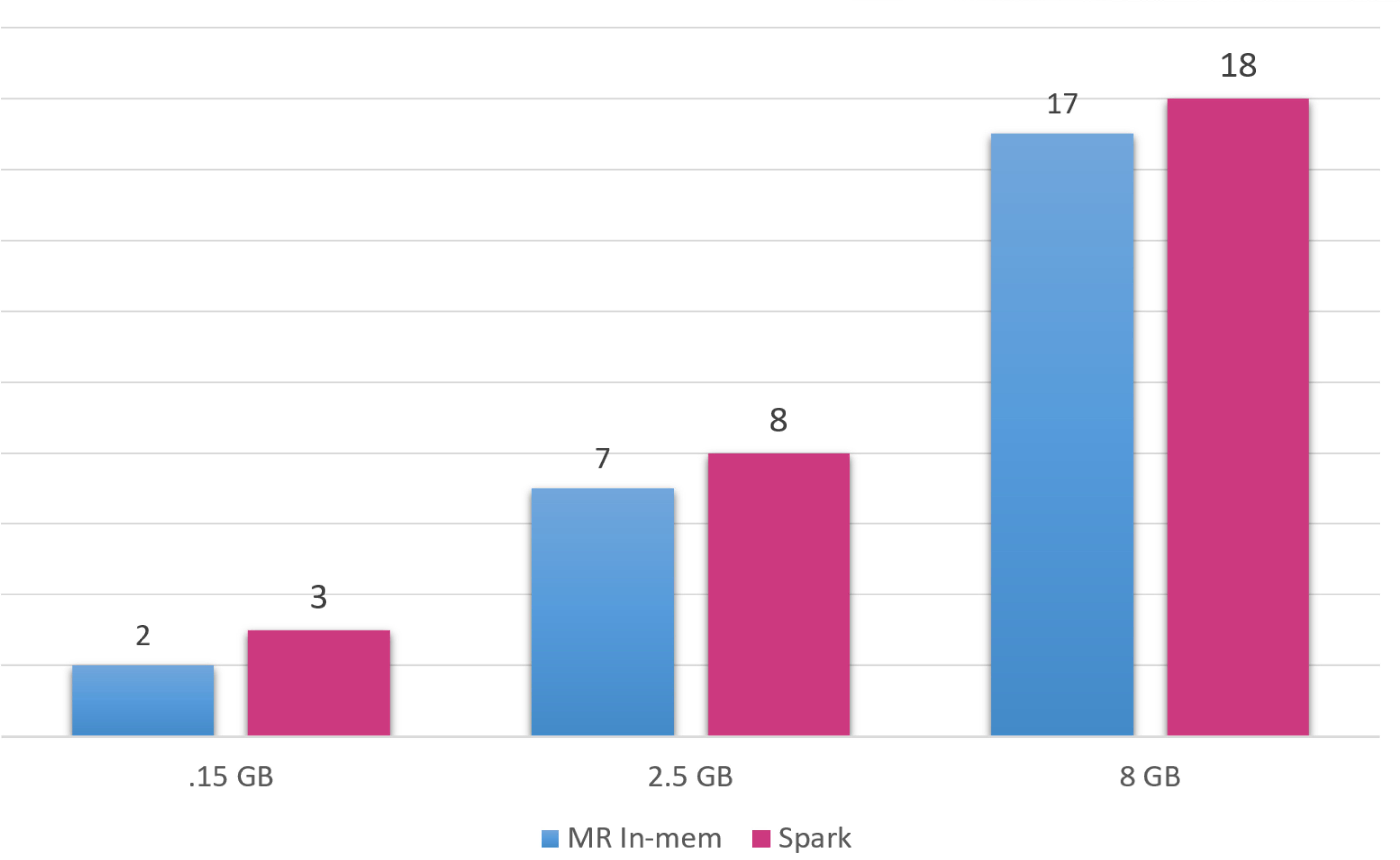


Kyligence

Spark Cubing vs. MR In-mem Cubing

Almost the same fast. And more adaptable to general data set.

- In-mem cubing expects sharded data, works poorly on random data sets.
- Spark cubing is more adaptable to different kinds of data distribution.

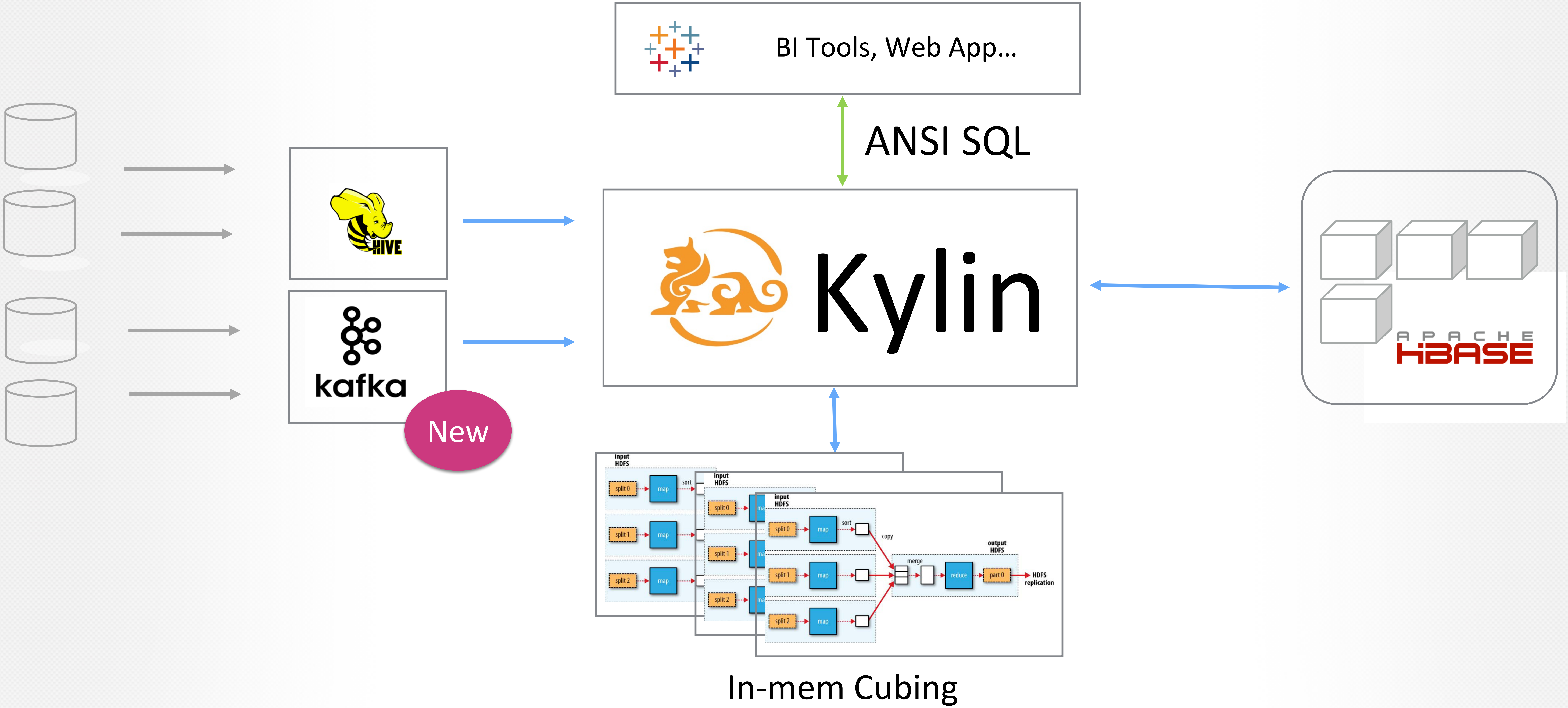


Kyligence

近实时流式构建

延时缩短到几分钟

Kylin 1.6 中的新功能



Kyligence

演示：Twitter 实时分析

http://hub.kyligence.io

增量构建每 2 分钟一次，每次耗时 3 分钟。

- 8-node cluster on AWS, 3 Kafka brokers
- Twitter sample feed, 每秒 10+ K 记录
- Cube 有 9 个维度和 3 个度量
- 2 个构建任务并发执行

Jobs in: LAST ONE WEEK ☐ NEW ☐ PENDING ☐ RUNNING ☐ FINISHED ☐ ERROR ☐ DISCARDED

Cube	Progress	Last Modified Time	Duration	Actions
embedded_cube	73.68%	2016-10-10 21:52:28 PST	1.65 mins	Action
embedded_cube		2016-10-10 21:52:23 PST	0.23 mins	Action
twitter_tag_cube2	100%	2016-10-10 21:44:19 PST	5.37 mins	Action
embedded_cube	100%	2016-10-10 21:44:14 PST	3.27 mins	Action
twitter_tag_cube2	100%	2016-10-10 21:38:15 PST	7.28 mins	Action
embedded_cube	100%	2016-10-10 21:37:38 PST	2.83 mins	Action
embedded_cube	100%	2016-10-10 21:34:26 PST	3.48 mins	Action
embedded_cube	100%	2016-10-10 21:24:14 PST	3.27 mins	Action

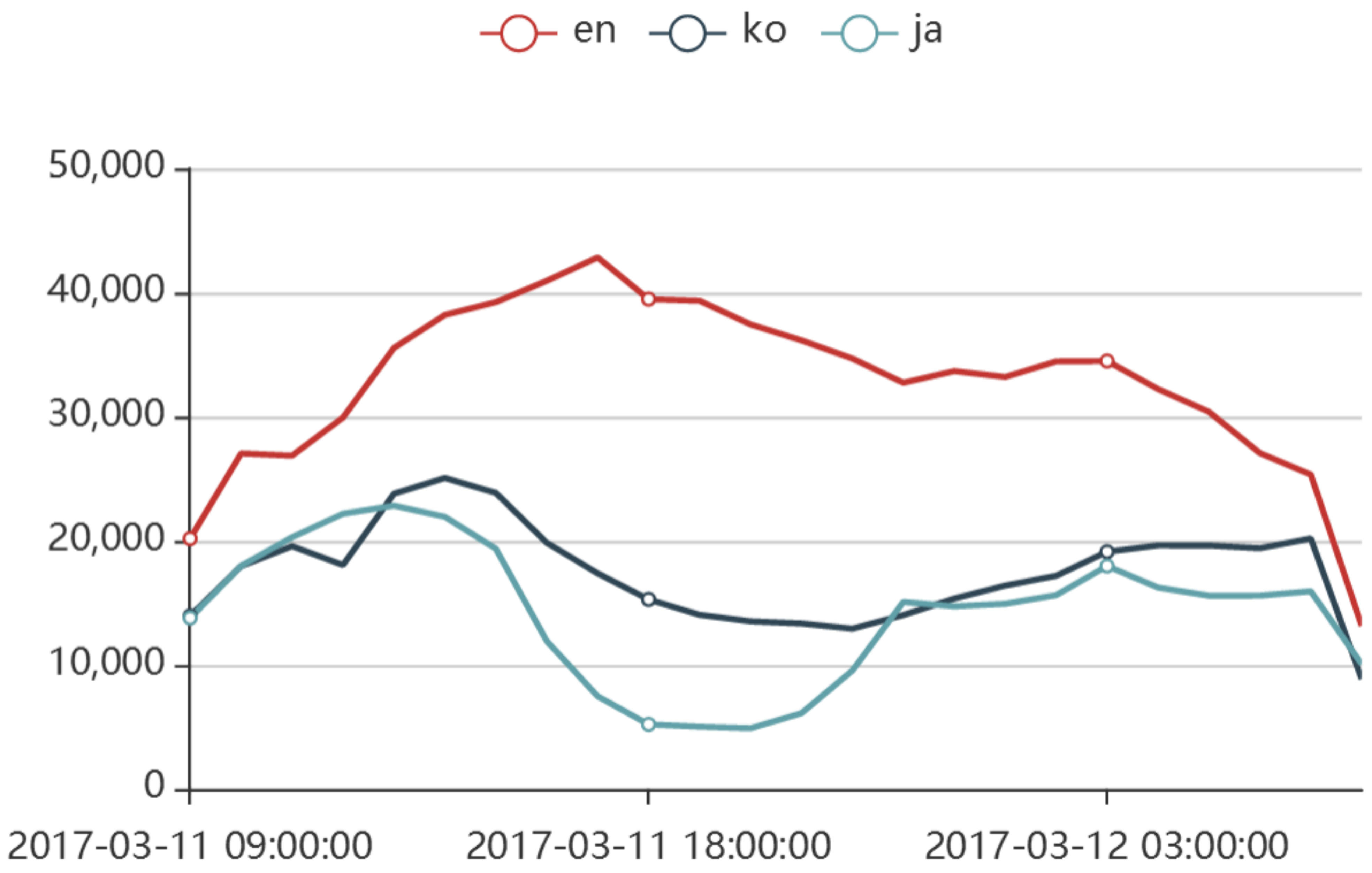
Real-Time Streaming Analytics for Twitter

请选择时间: Sat, 11 Mar 2017 09:12:59 GMT - Sun, 12 Mar 2017 09:12:59 GMT 每五分钟刷新一次

* 以当前时间选择,计算时自动转化为GMT格式时间

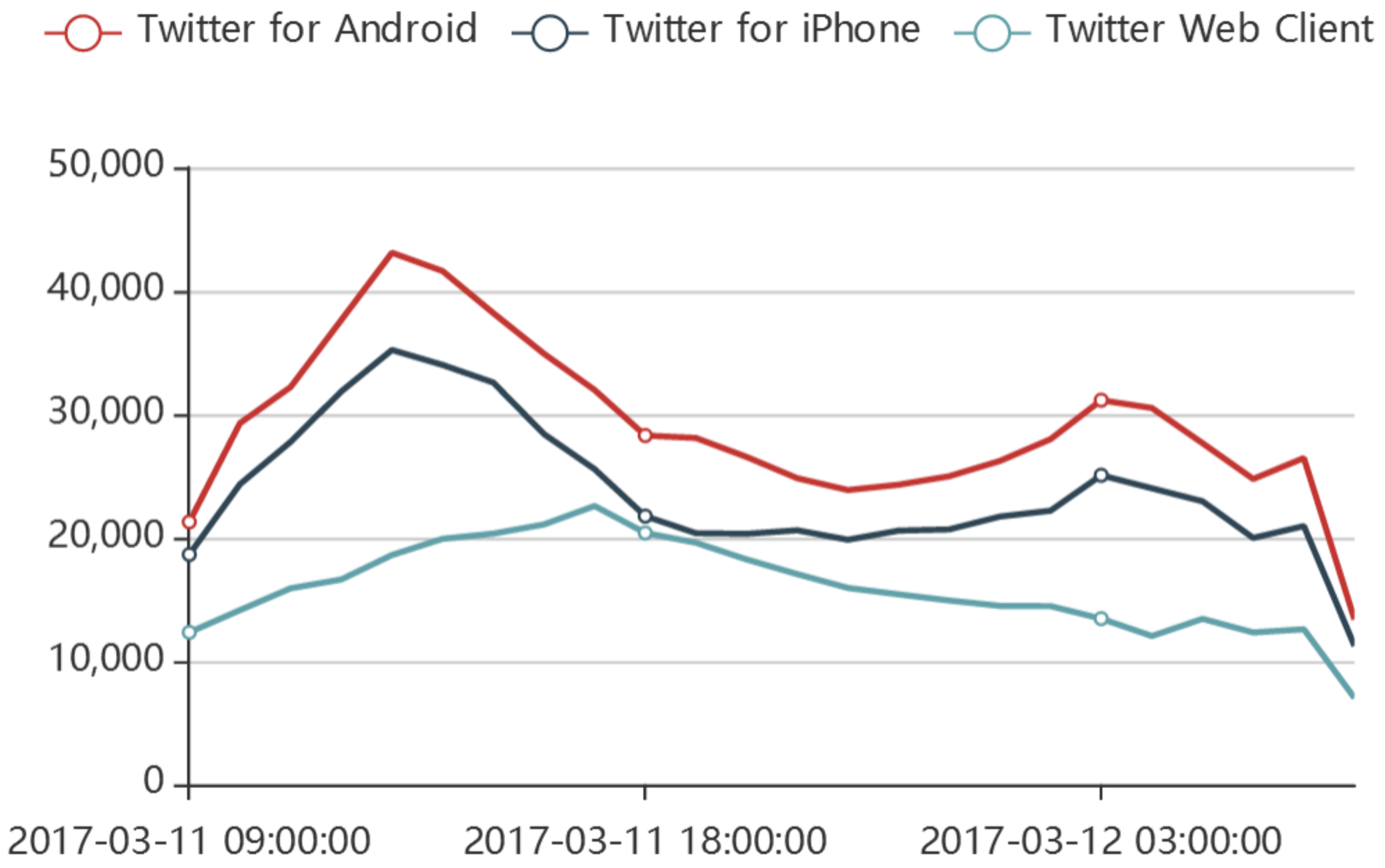
Language

en, ko, ja



Device

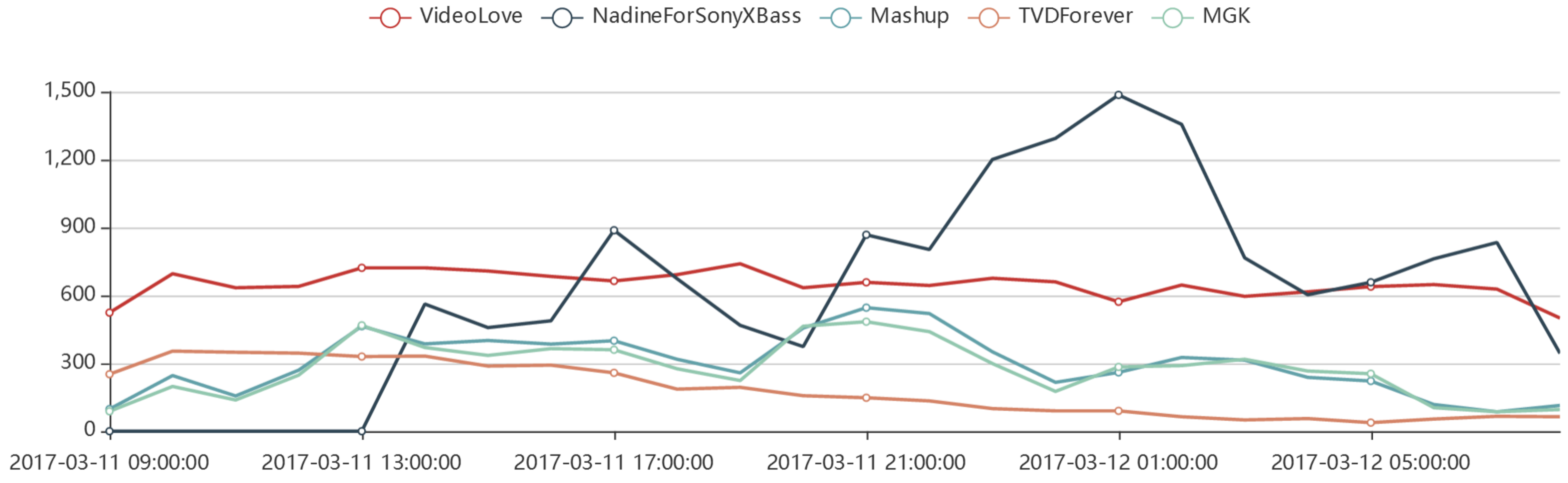
Twitter for Android, Twitter for iPhone, Twitter Web Client



en

Tag

VideoLove x | NadineForSonyXBass x | Mashup x | TVDForever x | MGK x



小结

Apache Kylin 2.0

- Kylin 2.0 Beta download 已经可以试用

- 支持雪花模型
- 运行 TPC-H 基准测试
- Time / Window / Percentile 函数

- Spark 构建引擎

- 近实时数据分析

What is next

- Hadoop 3.0 支持 (Erasure Coding)

- Spark 构建引擎持续改进

- 连通更多数据源 (JDBC, SparkSQL)

- 不同的存储 (Kudu?)

- 零延迟实时分析，Lambda 架构

Kyligence®

Unleash big data productivity

We are hiring!